Learning union of k-testable languages

Statistical and symbolic language modeling project

Rania el Bouhssini, Martin Laville and Félix Jamet

December 22, 2018

Contents

1 Introduction

Unless explicitely specified, all definitions and algorithms in this document are coming from Linard et al. [1].

We will present a possible implementation of those definitions and algorithms in a modular fashion, using Python3. Modular meaning here that we will implement concepts as they come and assemble them later as a whole when the necessary parts are complete. So if an `__init__` appears in the wild without its enclosing `class`, it's nothing to worry about.

2 k-test vector

A k-testable language can be described using a k-test vector which is a $4-tuple\ Z = \langle I, F, T, C \rangle$:

— $I \in \Sigma^{k-1}$ is a set of allowed prefixes,
— $F \in \Sigma^{k-1}$ is a set of allowed suffixes,
— $T \in \Sigma^{k}$ is a set of allowed segments, and
— $C \in \Sigma^{<k}$ is a set of allowed short strings satisfying $I \cap F = C \cap \Sigma^{k-1}$.

We will refer to I, F, T, C respectively as the allowed prefixes, suffixes, infixes and short strings. An intuitive way to formulate the constraint on short strings is that the short strings of length $k-1$ have to be both prefixes and suffixes and vice versa.

This definition can be translated into an init.

`Init k-test vector:`

```
def __init__(self, prefixes, suffixes, infixes, shorts):
    self.k = len(prefixes[0]) + 1
    self.prefixes = prefixes
    self.suffixes = suffixes
    self.infixes = infixes
    self.shorts = shorts
    self.ensure_correct_definition()
```

We then write `ensure_correct_definition` to make sure that the created k-test vector respects the conditions of the definition.

`Ensure correct definition:`

```
def ensure_correct_definition(self):
    def same_length(collection, reference_length):
        return all(filter(lambda x: len(x) == reference_length, collection))

    errors = []
    if not same_length(self.prefixes, self.k - 1):
        errors.append("Incorrect prefix length.")
    if not same_length(self.suffixes, self.k - 1):
        errors.append("Incorrect suffix length.")
    if not same_length(self.infixes, self.k):
        errors.append("Incorrect infix length.")
    if not all(filter(lambda x: len(x) < self.k, self.shorts)):
        errors.append("Incorrect short string length.")
    presufixes = self.prefixes & self.suffixes
    shorts_len_k = set(filter(lambda x: len(x) == self.k - 1))
    if presufixes != shorts_len_k:
        errors.append("Short strings conditions not satisfied.")

    if len(errors) >0:
        raise ValueError(' '.join(errors))
```

3 Sources

1. Linard, A., de la Higuera C., Vaandrager F.:Learning Unions of k-Testable Languages (`http://www.sws.cs.ru.nl/publications/papers/fvaan/kTestable/main.pdf`)